Using R Programming in a Financial Derivatives Course

Adam Y.C. Lei and Huihua Li[1]

ABSTRACT

R is a popular and free programming language and software environment for statistical computing and graphics. We use a class project in an undergraduate financial derivatives course to introduce students to R programming. Specifically, through pricing options using the Black-Scholes model and the multi-period binomial option pricing model, students learn to import and manipulate data in R, use R to do calculations, construct user-defined functions, and apply third-party packages in R. The inclusion of relevant technology through this project also serves to maintain our curriculum currency, and we obtain overwhelmingly positive student responses to the project.

Introduction

With the development of fintech and big data over the past decades, familiarity with programming languages has become a highly valuable skill for finance graduates (e.g., Dishman 2016). Even just the understanding of how programming works could provide graduates an edge over other job candidates. In this paper we illustrate a class project to introduce students to R programming in an undergraduate derivatives course. R is a free programming language and software environment for statistical computing and graphics. It is compatible with most computer operation systems and has thousands of free add-on packages that users can choose from. In 2019, R ranks 5th in job advertisements that highlight knowledge of data science software as a requirement, and 2nd in the most frequently used data science tools in scholarly articles (Muenchen 2020). As of September 2020, R ranks as the 9th most popular programming language (Wikipedia 2020).

In the class project, students use R to price options using the Black-Scholes model and the multi-period binomial option pricing model. Students learn not only the applications of option pricing models, but also the major characteristics of R programming and how to access R packages. The ability to employ third-party packages is one of R's most powerful functions. The open-source nature, the compatibility, and the abundance of third-party packages together distinguish R from other programming languages.[2] The R project not only allows our students to pursue more advanced applications using R, but also allows us to maintain the currency of our curriculum.

The paper is organized as follows: In the next section we review the background and structure of R programming language and related studies. The following section introduces our financial derivatives course and its organization. The fourth section illustrates the class project in detail. The fifth section discusses the fit of the project within our curriculum and student responses. The final section concludes and an appendix summarizes the specific R functions we use in this paper.

R and Related Literature

R is a free programming language and software environment that was initially developed by Ross Ihaka and Robert Gentleman at the University of Auckland in 1991, and it has since enjoyed growing popularity

[1] Adam Y.C. Lei, CFA, Dillard College of Business Administration, Midwestern State University, 3410 Taft Blvd, Wichita Falls, TX 76308, adam.lei@msutexas.edu. Huihua Li, CFA, Herberger Business School, St. Cloud State University, 720 4th Ave S, St. Cloud, MN 56301, hli@stcloudstate.edu. We thank Bill Yang and an anonymous referee for helpful comments.

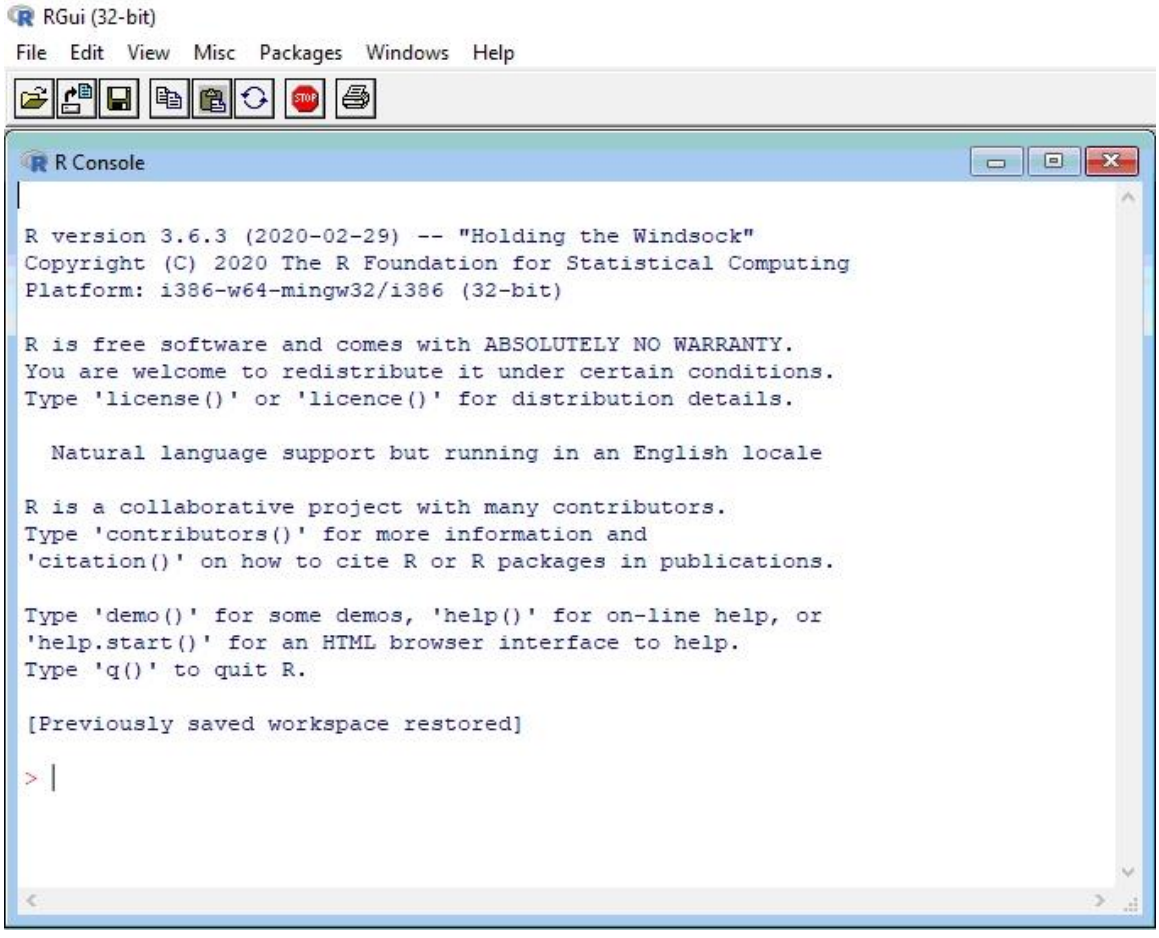
[2] Examples of other programming languages that have been used to price options include Java (e.g., Sedgewick and Wayne 2017), SAS (e.g., Clemmensen 2017), and Matlab (e.g., Zagaglia 2021). We thank an anonymous referee for pointing out those references.

Electronic copy available at: https://ssrn.com/abstract=3895539

(Wikipedia 2020). R currently has more than two million users worldwide and has an engaged community of programmers, educators, and users. Vries and Meys (2015) discuss several characteristics of R that could have contributed to its success and popularity: First, R is a powerful statistical programming and graphic tool. It can perform a variety of functions in areas such as linear/nonlinear modeling, statistical tests, time series analysis, graphics, data manipulation, and data visualizations. Second, R is open source and is free to install and use. This characteristic attracts expert programmers to continuously contribute to the maintenance and improvement of R, making it more stable and reliable over time. Third, R can run in most operating systems including Windows, MacOS, and Unix, is highly extensible, and has thousands of free add-on packages that users can choose from. The abundance of free third-party packages makes R even more attractive and convenient for its users. Finally, R can easily connect with other programming languages such as SAS and SPSS, and with finance databases such as Bloomberg.

In practice, many users use R within the RStudio, an integrated development environment for R that offers a richer and easier editing environment. Users can download R from *https://www.r-project.org/*, and RStudio from *https://rstudio.com/products/rstudio/download/*. Once a user enters R, the starting screen (the R Console) lists its basic information such as version, license, and how to access online help and demo. The line starting with a symbol ">" indicates where the user can write codes/commands in R. Figure 1 shows the starting screen of R.

Figure 1: Starting Screen of R

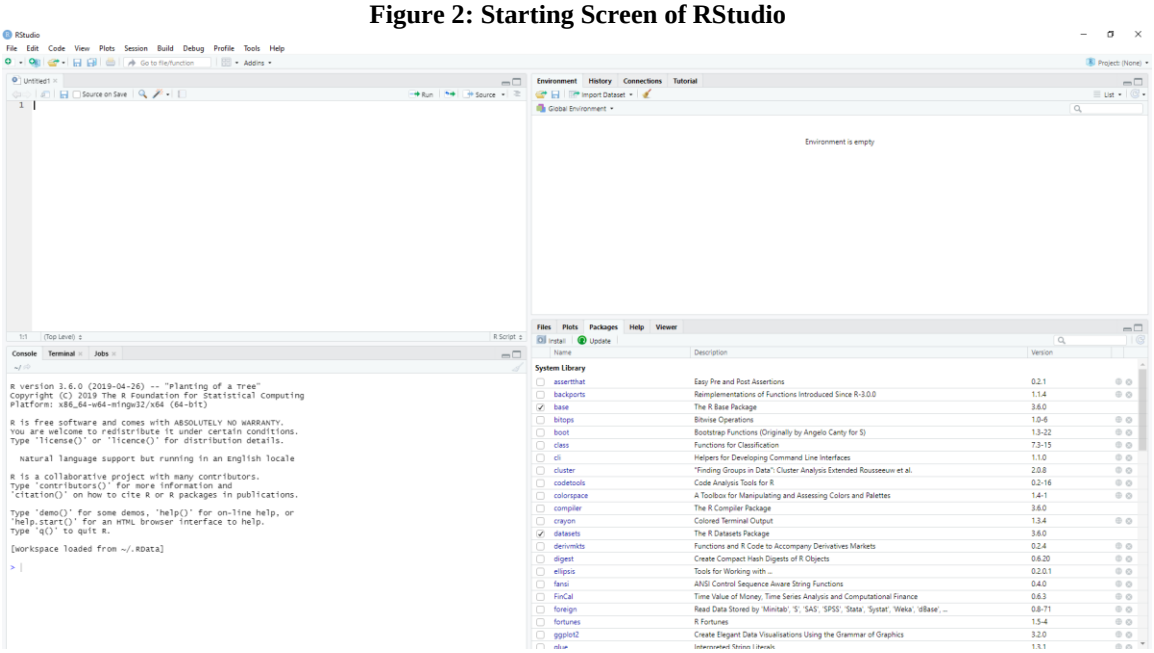

RGui (32-bit)
File Edit View Misc Packages Windows Help
R Console
R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Previously saved workspace restored]
>

In the starting screen of RStudio, there are four blocks. The upper-left block is Source, a text editor. In this text editor block, a user can enter/edit codes, save the codes, and work with other source script files (the codes are also called scripts in R). The lower-left block is Console, which shows all the interactive work with R. This part can be used the same way as the R Console in R. The upper-right block is Environment and History, in which a user can view the variables that have been created in the sessions and the history of the

Electronic copy available at: https://ssrn.com/abstract=3895539

commands that the user has issued. The lower-right block shows Files, Plots, Package, Help, and Viewer. Through this block a user can browse existing files, view the plots, view the list of packages installed/available, access the built-in help, and view local web content (e.g., web graphics generated by packages). Figure 2 shows the starting screen of RStudio.

Figure 2: Starting Screen of RStudio


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1
Source on Save
Run
Source
Environment History Connections Tutorial
Import Dataset
Global Environment
Environment is empty
Console Terminal Jobs
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
Files Plots Packages Help Viewer
Install Update
Name Description Version
System Library
assertthat Easy Pre and Post Assertions 0.2.1
backports Reimplementations of Functions Introduced Since R-3.0.0 1.1.4
base The R Base Package 3.6.0
bitops Bitwise Operations 1.0-6
boot Bootstrap Functions (Originally by Angelo Canty for S) 1.3-22
class Functions for Classification 7.3-15
cli Helpers for Developing Command Line Interfaces 1.1.0
cluster "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. 2.0.8
codetools Code Analysis Tools for R 0.2-16
colorspace A Toolbox for Manipulating and Assessing Colors and Palettes 1.4-1
compiler The R Compiler Package 3.6.0
crayon Colored Terminal Output 1.3.4
datasets The R Datasets Package 3.6.0
derivmkts Functions and R Code to Accompany Derivatives Markets 0.2.4
digest Create Compact Hash Digests of R Objects 0.6.20
ellipsis Tools for Working with ... 0.2.0.1
fansi ANSI Control Sequence Aware String Functions 0.4.0
FinCal Time Value of Money, Time Series Analysis and Computational Finance 0.6.3
foreign Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ... 0.8-71
fortunes R Fortunes 1.5-4
ggplot2 Create Elegant Data Visualisations Using the Grammar of Graphics 3.2.0

Despite the popularity of R in practice, only a few academic studies address the uses of R in business disciplines, and virtually none focuses on the pedagogical applications of R. For instance, Allen (2017) and Ozgur et al. (2018) provide an introduction and overview of the R programming language. Das (2014) and Krotov and Tennyson (2018) use R for text extraction and web scrapping to obtain data. Turner (2015) provides the R codes to calculate the weights for optimal stock portfolios. Gallagher and Trendafilov (2018) compare and contrast the uses of R and Python on basic statistical analyses. Our paper linking the use of R with an undergraduate financial derivatives course through a class project thus fills a gap in the literature.

The Financial Derivatives Course

Our financial derivatives course serves as an upper-level elective for finance majors and requires the business/managerial finance course as a prerequisite (completion of the investments course is preferred). Per the course description, this course examines the "characteristics and functions of financial derivatives, corporate risk management applications of financial derivatives; and pricing models of derivatives and trading strategies using derivatives to hedge financial risks." We allocate 75% of the course content to address options and their applications, including the structure of options market, principles of option pricing, option pricing models, and option strategies. The remaining 25% of the course content addresses futures, forwards and swaps, and their applications in hedging and risk management.

Given that this course does not have much content overlap with other finance courses, our emphasis foremost is on course-specific knowledge. We additionally aim to address practical applications of the course content and to incorporate relevant technology. Following this philosophy, we assign 55% of the semester grade on the assessment of a student's course-specific knowledge (5% on quizzes, 15% on the first midterm exam, 15% on the second midterm exam, and 20% on the final exam). We assign 15% of the semester grade on a student's individual assignments, mostly end-of-chapter problems, to ensure that students have enough practice to digest the course material. 10% of the semester grade depends on a student's class participation. The remaining 20% is based on two team-based projects, with each accounting for 10% of the semester grade. Students form teams of four or five students, depending on the class size, for the team-based projects.

Electronic copy available at: https://ssrn.com/abstract=3895539

The first team project requires students to manage a simulated portfolio in Stock-Trak using options, futures, and stocks (e.g., Lei and Li 2012). Each team reports its trading activities and justification on a weekly basis and submits a report detailing its takeaways at the end of the semester. This semester-long project allows students to apply their knowledge and gain practical experience. We grade this project based on whether a team executes and justifies its trades over time as required, and the quality of its semester-end report.

The second team project, through which we incorporate the use of R, is a computer-based option pricing project. In this project, we require students to use both R programming and Excel to price European options of a stock the team chooses, using the Black-Scholes model and the multi-period binomial option pricing model. Each team starts with the collection of the underlying stock prices and the risk-free rate for the project. Option pricing, an important component for the financial derivatives course, involves complicated calculations and can be intimidating to students. Using both R programming and Excel to price options help students solidify their understanding of the pricing models. Each team eventually submits its inputs and outputs in R and in Excel worksheets for grading, and we grade the project based on the correctness of the inputs and the accuracy of the outputs. We provide further details of this project (the R project) in the following section.

The R Project

Our R project requires students in teams to use R (and Excel) to import data, calculate stock returns and return standard deviation, and price a European call option and a European put option of a stock they choose, using the Black-Scholes model and the multi-period binomial option pricing model. Specifically, students are required to: 1) choose a non-dividend paying stock and one European call option and one European put option of the stock to value. The expiration date of the options should be around one to three months and the options should not be deep in or deep out of the money (so the option value would not be predominantly from the time value or the intrinsic value), 2) obtain the daily stock prices from Yahoo! Finance over the most recent 12 months and calculate the stock returns and return standard deviation, in addition to obtaining the corresponding risk-free rate from the Treasury security with the maturity closest to the option expiration date from the Wall Street Journal *(https://www.wsj.com/market-data/bonds/treasuries*), 3) use R programming (and Excel) to calculate the option values using the Black-Scholes model, 4) use R programming (and Excel) to calculate the option values using the multi-period binomial option pricing model.

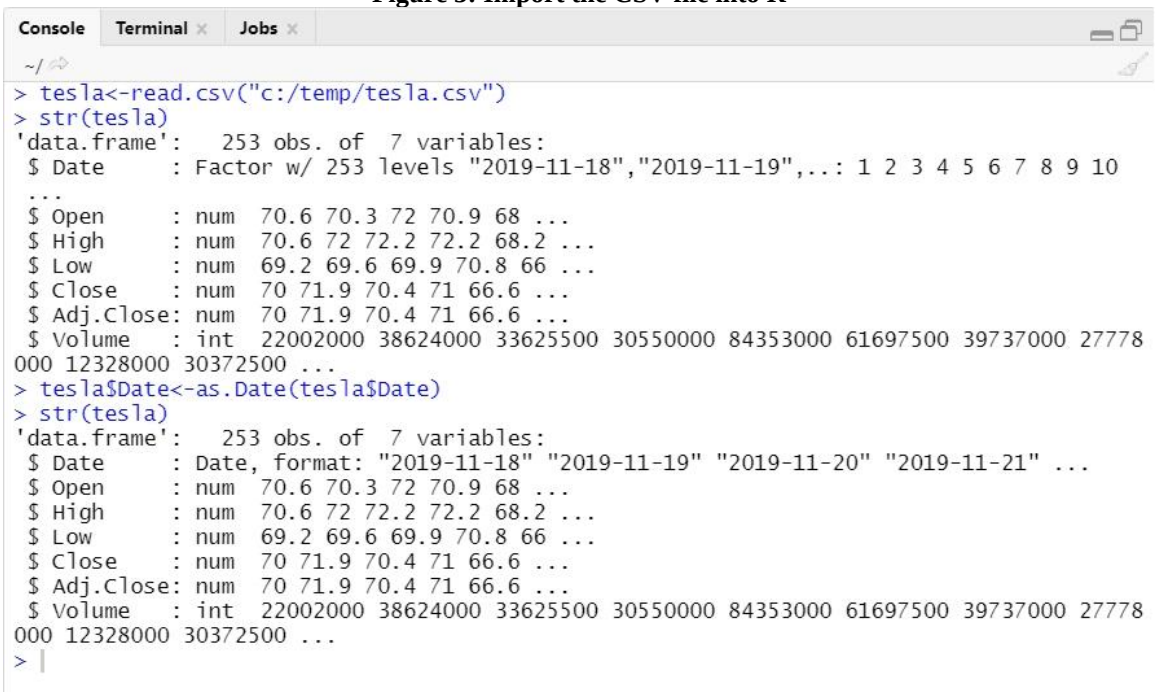
Given the inexperience of our students with R programming, we illustrate in class the key commands and functions of R that students need to understand for this project. We also spend some, but less time discussing the relevant Excel functions for this project, since most students already have some degree of familiarity with Excel. Below we illustrate the step-by-step procedures to implement the project.

Data Import and Stock Returns/Return Standard Deviation Calculation

In our illustration we use the stock of Tesla, Inc. (TSLA) as an example for the underlying (non-dividend paying) stock. The daily stock price data we download from Yahoo! Finance are in a file of the CSV format, and in R we use "*read.csv()"* to import the CSV file. We download the most recent one-year daily stock price data of Telsa (from November 18, 2019 to November 17, 2020) and save the file as *"c:\temp\tesla.csv"*. We use the R command *"tesla<-read.csv("c:/temp/tesla.csv")"* to import the CSV data into a data frame named *"tesla"*, where the data frame is effectively a dataset in R. One caveat is that, in R, to list the location of the file we need to replace "\" with "/" (i.e., instead of using *"c:\temp\tesla.csv"* we use *"c:/temp/tesla.csv"*). After the import, we can use the R function *"str()"* to display the structure of the data frame *tesla*, referred to within the parentheses (i.e., *"str(tesla)"*). The variable *Date*, however, is not imported in the correct date format at this time. To convert it to the date format, we use the function *"as.Date()"* (the "*Date"* must be capitalized). Specifically, in the command *"tesla$Date<-as.Date(tesla$Date)"*, *tesla$Date* refers to the variable of interest, i.e., the *Date* variable in the *tesla* data frame. We can then verify the structure of the data frame again using the function *"str()"*, and see that the *Date* variable is now in the correct date format. Figure 3 shows the implementation of these steps.

Electronic copy available at: https://ssrn.com/abstract=3895539

Figure 3: Import the CSV file into R



```
Console  Terminal  Jobs
~/
> tesla<-read.csv("c:/temp/tesla.csv")
> str(tesla)
'data.frame':   253 obs. of  7 variables:
 $ Date     : Factor w/ 253 levels "2019-11-18","2019-11-19",..: 1 2 3 4 5 6 7 8 9 10
 ...
 $ Open     : num  70.6 70.3 72 70.9 68 ...
 $ High     : num  70.6 72 72.2 72.2 68.2 ...
 $ Low      : num  69.2 69.6 69.9 70.8 66 ...
 $ Close    : num  70 71.9 70.4 71 66.6 ...
 $ Adj.Close: num  70 71.9 70.4 71 66.6 ...
 $ Volume   : int  22002000 38624000 33625500 30550000 84353000 61697500 39737000 27778
000 12328000 30372500 ...
> tesla$Date<-as.Date(tesla$Date)
> str(tesla)
'data.frame':   253 obs. of  7 variables:
 $ Date     : Date, format: "2019-11-18" "2019-11-19" "2019-11-20" "2019-11-21" ...
 $ Open     : num  70.6 70.3 72 70.9 68 ...
 $ High     : num  70.6 72 72.2 72.2 68.2 ...
 $ Low      : num  69.2 69.6 69.9 70.8 66 ...
 $ Close    : num  70 71.9 70.4 71 66.6 ...
 $ Adj.Close: num  70 71.9 70.4 71 66.6 ...
 $ Volume   : int  22002000 38624000 33625500 30550000 84353000 61697500 39737000 27778
000 12328000 30372500 ...
>
```

After importing the data, we can view the content of the data frame using the function *"View()"* (i.e., *"View(tesla)"*; this function must be capitalized). The data frame *tesla* has seven variables: *Date, Open Price, High Price, Low Price, Close Price, Adjusted Close Price,* and *Volume*. Figure 4 shows the sample content of the *tesla* data frame.

Figure 4: View of Data File *tesla*

Filter

	Date	Open	High	Low	Close	Adj.Close	Volume
1	2019-11-18	70.584	70.630	69.220	69.998	69.998	22002000
2	2019-11-19	70.350	71.998	69.560	71.904	71.904	38624000
3	2019-11-20	72.000	72.240	69.914	70.444	70.444	33625500
4	2019-11-21	70.902	72.168	70.800	70.966	70.966	30550000
5	2019-11-22	68.032	68.200	66.000	66.608	66.608	84353000
6	2019-11-25	68.864	68.914	66.892	67.268	67.268	61697500
7	2019-11-26	67.054	67.100	65.420	65.784	65.784	39737000
8	2019-11-27	66.224	66.786	65.714	66.258	66.258	27778000
9	2019-11-29	66.222	66.252	65.500	65.988	65.988	12328000
10	2019-12-02	65.880	67.276	65.738	66.974	66.974	30372500
11	2019-12-03	66.524	67.582	66.438	67.240	67.240	32868500

To calculate the stock returns, we need the two variables *Date* and *Adjusted Close Price* from the original data frame *tesla*. We use the command *"tesla1<-tesla[,c("Date","Adj.Close")]"* to create a new data frame

Electronic copy available at: https://ssrn.com/abstract=3895539

tesla1 containing those two variables. We then create a new variable *price1* to represent the one-day lag of the *Adjusted Close Price*, i.e., the adjusted close price of the stock on the previous trading day. Specifically, we use the R functions *"c()", "head()",* and *"cbind()"* to create a new data frame *tesla2* that combines the data frame *tesla1* and the lagged variable *price1*. The R commands we use are *"price1<-c(NA,head(tesla1$Adj.Price,-1))"* and *"tesla2<-cbind(tesla1,price1)".* In this case "*head(tesla1$Adj.Price,-1)"* retains all but the last row of the variable *Adjusted Close Price* in the data frame *tesla1*, and *"c(NA,head(tesla1$Adj.Price,-1)"* creates the one-day lag of the *Adjusted Close Price* by adding a missing value *"NA"* as the first observation. *"cbind(tesla1,price1)"* combines the data frame *tesla1* and the lagged variable *price1*. We then calculate the daily stock return (variable *return*) on day *t* as the natural log of (adjusted close price on day *t*/adjusted close price on day $t - 1$) and create a new data frame *tesla3* that combines the data frame *tesla2* and the *return* variable, using the commands *"return<-log(tesla2$Adj.Price/tesla2$price1)"* and *"tesla3<cbind(tesla2,return)".* We use the function *"sd()"* to obtain the daily return standard deviation (variable *std*), and annualize it (variable *annualstd*) by multiplying the daily return standard deviation (*std*) with the square root of 252 (trading days per year), using the function *"sqrt()"*. The R commands we use are respectively *"std<-sd(tesla3$return,na.rm=TRUE)"* and *"annualstd<-std*sqrt(252)"* (the argument *"na.rm=TRUE"* in the *"sd()"* function removes missing values, if any, from the calculation of the standard deviation). Once the variables are created, a command of the variable name shows the value of the variable (i.e., *std* = 0.05538341 and *annualstd* = 0.8791844 in this example). Figure 5 shows the calculations.

Figure 5: Return and Return Standard Deviation Calculations

Import data and calculating return and ...* | tesla3

Filter

	Date	Adj.Close	price1	return
1	2019-11-18	69.998	NA	NA
2	2019-11-19	71.904	69.998	0.026865198
3	2019-11-20	70.444	71.904	-0.020513814
4	2019-11-21	70.966	70.444	0.007382863
5	2019-11-22	66.608	70.966	-0.063376211
6	2019-11-25	67.268	66.608	0.009859890
7	2019-11-26	65.784	67.268	-0.022308009
8	2019-11-27	66.258	65.784	0.007179656
9	2019-11-29	65.988	66.258	-0.004083365

Showing 1 to 9 of 253 entries, 4 total columns

Console | Terminal | Jobs

~/

```
> 
> #arrange the data
> tesla1<-tesla[,c("Date","Adj.Close")]
> price1<-c(NA,head(tesla1$Adj.Close,-1))
> tesla2<-cbind(tesla1,price1)
> return<-log(tesla2$Adj.Close/tesla2$price1)
> tesla3<-cbind(tesla2,return)
> 
> #calculate the annualized standard deviation
> std<-sd(tesla3$return,na.rm=TRUE)
> std
[1] 0.05538341
> annualstd<-std*sqrt(252)
> annualstd
[1] 0.8791844
```


Electronic copy available at: https://ssrn.com/abstract=3895539

Using the Black-Scholes Model

Black and Scholes (1973) and Merton (1973) develop the widely used model, i.e., the Black-Scholes model, to price European options on non-dividend paying stocks. Although later the model was extended to price European options on dividend-paying stocks, we use the original model in our project for its simplicity. Specifically, the Black-Scholes model prices a European call option on a non-dividend paying stock as follows:

$$C_0 = S_0 N(d_1) - X e^{-rT} N(d_2)$$

where

$$d_1 = \frac{\ln(S_0/X) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_2 = d_1 - \sigma\sqrt{T}$$

and where

C_0 = Current call option value.
S_0 = Current stock price.
$N(d)$ = The probability that a random draw from a standard normal distribution will be less than d.
X = Exercise price.
e = Base of the natural log function.
r = Risk-free rate (annualized continuously compounded).
T = Time until expiration of option in years.
σ = Annualized standard deviation of the continuously compounded stock return.

Using the put-call parity, the Black-Scholes model prices a European put option on a non-dividend paying stock as follows:

$$P_0 = X e^{-rT}[1 - N(d_2)] - S_0[1 - N(d_1)]$$

where

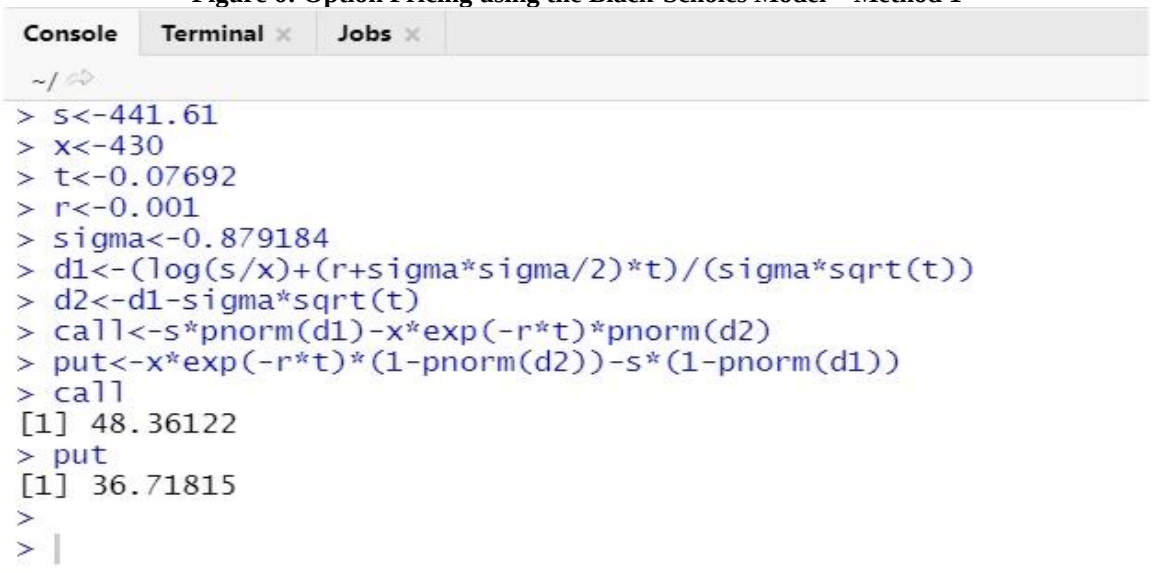
P_0 = Current put option value, and the other variables are as defined before.

To help students better understand R programming, we illustrate two methods in R to find the Black-Scholes option values. The first method effectively uses R as a calculator, and the second method involves a user-defined function. We use European options on the non-dividend paying stock of Tesla as an example: The stock price on November 17, 2020 is \$441.61. The call option and put option we intend to price expire on December 18, 2020 (in about 4 weeks, T = 4/52 = 0.07692) and have an exercise of \$430. The annualized standard deviation is 0.879184 as previously calculated, and the annualized yield of the Treasury security with the maturity date closest to December 18, 2020 is 0.10%.

Figure 6 illustrates the first method. Specifically, we assign the value to each input variable, including stock price *s* = 441.61, exercise price *x* = 430, time to expiration *t* = 0.07692, risk-free rate *r* = 0.001, and return standard deviation *sigma* = 0.879184, by using *"<-"* in R. We then explicitly define d1, d2, and the call and put option values in R using the Black-Scholes model. In this later step, we also use the R functions *"log()"* for the natural log, *"sqrt()"* for the square root, *"pnorm()"* for the cumulative normal distribution function, and *"exp()"* for the power of *e*. The calculated call option value is \$48.36122 and the put option value is \$36.71815.

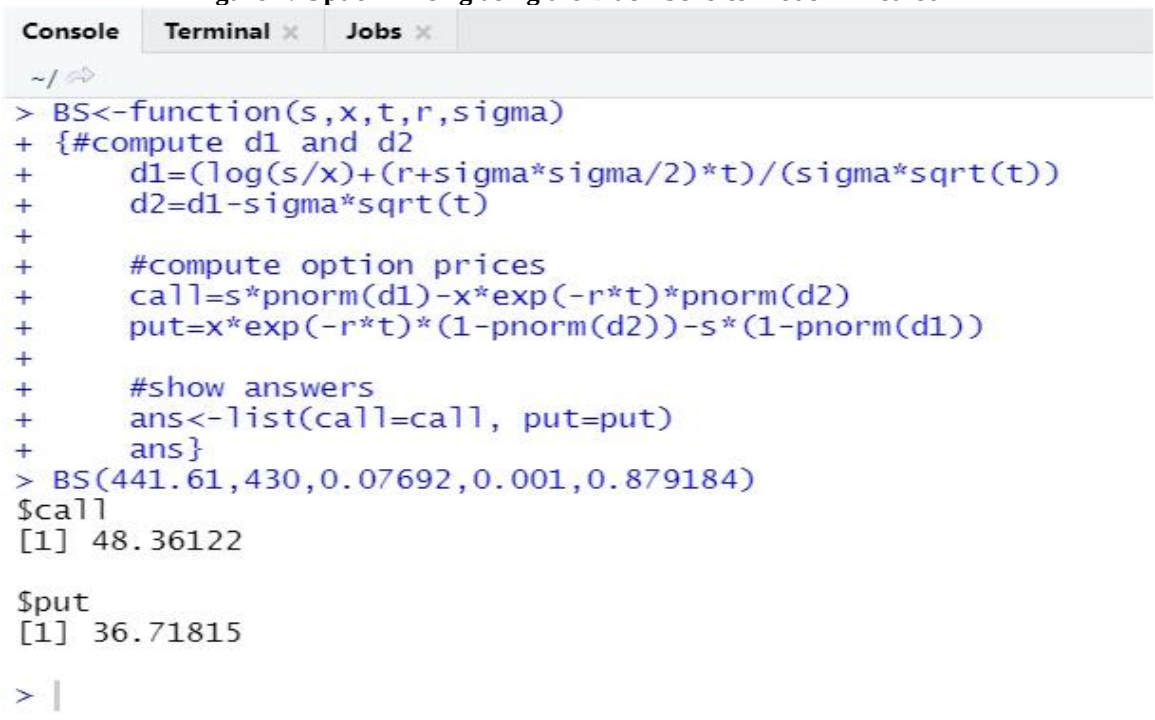
Electronic copy available at: https://ssrn.com/abstract=3895539

Figure 6: Option Pricing using the Black-Scholes Model – Method 1



The second method using the Black-Scholes model requires a user-defined function, and Figure 7 illustrates this method. Specifically, the user-defined function follows the format *"name <- function(arguments) {commands}"*. We define the BS function to calculate the call and put option values using the Black-Scholes model as *"BS<-function(s,x,t,r,sigma) {#compute d1 and d2, commands; #compute option prices, commands; #show answers, commands}"*(lines starting with # in R shows explanatory comments). The *"list()"* function within creates a list of the call option value and the put option value. After defining the BS function, we run the function script in the RStudio Console and input the variables in the format of *"BS(s,x,t,r,sigma)"* to obtain the option values. Compared to the first method, the second method with the user-defined function is more flexible in that we can easily price a different option by changing the inputs.

Figure 7: Option Pricing using the Black-Scholes Model – Method 2



Electronic copy available at: https://ssrn.com/abstract=3895539

Using the Multi-period Binomial Option Pricing Model

It is challenging for beginners to directly write codes in R for the multi-period binomial option pricing model. Therefore, we introduce the third-party packages concept to students as an alternative. R has a large number of third-party packages that a user can install for specific purposes. The packages are available on the Comprehensive R Archive Network (CRAN) at *https://cran.r-project.org/*. For our purpose, we search for the term "derivatives" on CRAN and use the package "derivmkts: Functions and R Code to Accompany Derivatives Markets" (*https://cran.r-project.org/package=derivmkts*) for the application of the multi-period binomial option pricing model.

To use a third-party package, we first need to install the package in R using the command *"install.packages ("package-name")"* (e.g., *"install.packages("derivmkts")"*). After installing the package, we use the command *"library ("package-name")"* (e.g., *"library("derivmkts")"*) to load it and make it ready to use. We remind students that each time when they restart RStudio, they need to load the package again before using it.

From the manual of the *"derivmkts"* package (McDonald 2019), we find that the function *"binomopt()"* uses the multi-period binomial option pricing model to compute the prices of European and American call options and put options. The detailed specification of the function is as follows:

binomopt(s, k, v, r, tt, d, nstep=10, american=TRUE, putopt=FALSE,
specifyupdn=FALSE, crr=FALSE, jarrowrudd=FALSE, up=1.5, dn=0.5,
returntrees=FALSE, returnparams=FALSE, returngreeks=FALSE)

For this function, the required inputs include the following six variables:

s = Current stock price.
k = Exercise price.
v = Annualized standard deviation of the continuously compounded stock return.
r = Risk-free interest rate (annualized continuously compounded).
tt = Time until expiration of option in years.
d = Dividend yield (annualized continuously compounded).

The other arguments that we use for our project include:

nstep = Number of binomial steps. Default is *nstep* = 10.
american = Indicate whether the option is an American option. Default is *american* = TRUE.
putopt = Indicate whether the option is a put option. Default is *putopt* = FALSE.
returntrees = Indicate whether to return four trees in the output: the price of the underlying asset (stree), the option price (oppricetree), where the option is exercised (exertree), and the probability of being at each node (probtree). If TRUE, this argument overrides the default *returnparams*= FALSE and *returngreeks* = FALSE arguments below. Default is *returntrees* = FALSE.

The arguments that we do not specify or change their defaults include:

specifyupdn = Indicate whether to manually enter the binomial parameters *up* and *dn* below. If TRUE, this argument overrides the *crr* and *jarrowrudd* arguments below. Default is *Specifyupdn* = FALSE.
crr = Indicate whether to use the Cox-Ross-Rubinstein tree. Default is *crr* = FALSE.
jarrowrudd = Indicate whether to use the Jarrow-Rudd tree. Default is *jarrowrudd* = FALSE.
up, dn = The specified up and down moves on the binomial tree if *specifyupdn* = TRUE.
returnparams = Indicate whether to return the vector of inputs and computed pricing parameters as well as the price. Default is *returnparams* = FALSE.
returngreeks = Indicate whether to return time 0 delta, gamma, and theta in the vector greeks. Default is *returngreeks* = FALSE.

Additionally, the function *"binomopt ()"* defines the risk-free rate per period (*rp*), the per-period up (*up*) and down (*dn*) parameters, and the binomial (risk-neutral) probability (*p*) as follows:

Electronic copy available at: https://ssrn.com/abstract=3895539

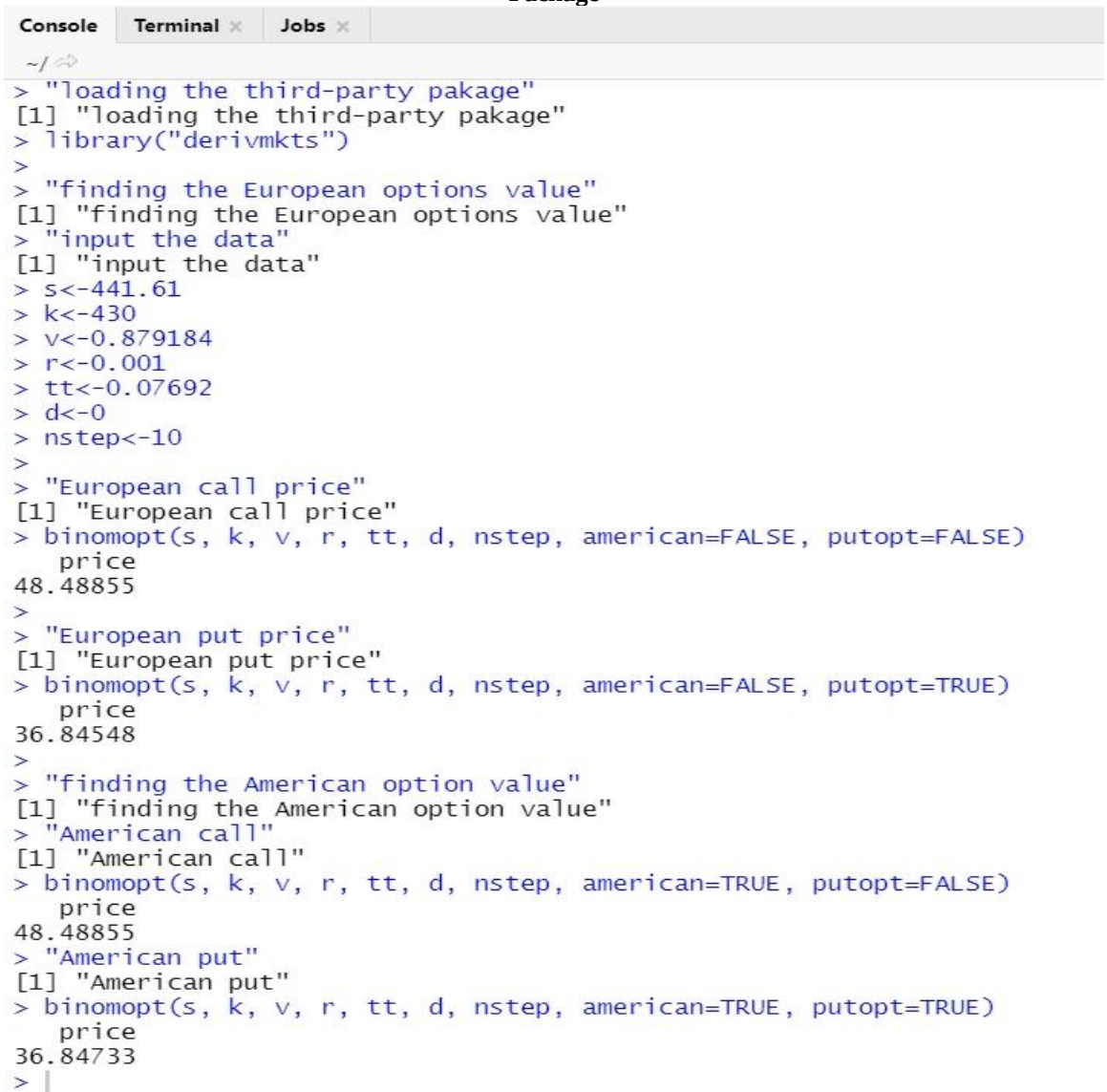
$h = tt/\#periods$
risk-free rate per period (rp) = $\exp(r \times h) - 1$
$up = \exp\left[(r - d) \times h + v\sqrt{h}\right]$
$dn = \exp\left[(r - d) \times h - v\sqrt{h}\right]$
binomial (risk-neutral) probability (p) = $(1 + rp - dn)/(up - dn)$

where

 # periods = Number of periods (steps), and the other variables are as defined before.

Using the same option inputs on the non-dividend paying stock of Tesla as an example, Figure 8 illustrates the step-by-step procedures using the multi-period binomial option pricing model with the derivatives package. Specifically, after we load the package using the command *"library("derivmkts")"*, we input the values for the required six variables (i.e., s = 441.61, k = 430, v = 0.879184, r = 0.001, tt = 0.07692, and d = 0) and specify the number of steps at its default = 10. We later require students to try 100 instead of 10 steps.

Figure 8: Option Pricing using the Multi-period Binomial Option Pricing Model with a Derivatives Package



Electronic copy available at: https://ssrn.com/abstract=3895539

By calling up the function *"binomopt ()"*, we first calculate the value of the European call option (i.e., *"american=FALSE, putopt=FALSE"*) and the value of the European put option (i.e., *"american=FALSE, putopt=TRUE"*). The value of the European call (put) option is $48.48855 ($36.84548) given the 10 binomial steps. We then calculate the value of an American call option (i.e., *"american=TRUE, putopt=FALSE"*) and the value of the American put option (i.e., *"american=TRUE, putopt=TRUE"*) with the same input variables. The value of the American call (put) option is $48.48855 ($36.84733) given the 10 binomial steps. The identical value of the American call option and the European call option, and the larger value of the American put option than the European put option, are consistent with theory that an American call option on a non-dividend paying stock has the same value as a European call option, and that an American put option is more valuable than a European put option (see Black and Scholes, 1973, and Merton, 1973).

Finally, in illustrating the procedures, we also use the *"returntrees=TRUE"* argument in the function *"binomopt()"* to show the greek variables (*greeks*), the input variables and the computed pricing parameters (*params*), the option price tree (*oppricetree*), and the stock price tree (*stree*), among other outputs. For brevity, Figure 9 shows the partial output of the *"binomopt()"* function with the *"returntrees=TRUE"* argument for the European call option.

Figure 9: European Call Option Tree and Underlying Stock Price Tree

```
Console  Terminal  Jobs
~/
> binomopt(s, k, v, r, tt, d, nstep,american=FALSE,returntrees=TRUE)
$price
   price
48.48855

$greeks
       delta        gamma        theta
 0.590714202  0.003796474 -0.784950332

$params
          s           k           v           r          tt           d       nstep           p          up          dn
441.6100000 430.0000000   0.8791840   0.0010000   0.0769200   0.0000000  10.0000000   0.4807325   1.0801670   0.9257970
          h
  0.0076920

$oppricetree
          [,1]     [,2]     [,3]       [,4]       [,5]       [,6]       [,7]       [,8]       [,9]      [,10]      [,11]
 [1,] 48.48855 69.39971 96.74480 131.052819 172.206169 219.3884306 271.4433399 327.671602 388.407782 454.013266 524.87841
 [2,]  0.00000 29.12993 44.08494  64.984210  92.955405 128.5278600 171.1997729 219.391806 271.447516 327.676643 388.41376
 [3,]  0.00000  0.00000 15.28516  24.737267  39.089723  60.0241639  89.0245393 126.586613 171.202407 219.395181 271.45169
 [4,]  0.00000  0.00000  0.00000   6.534715  11.450276  19.7094089  33.1767994  54.251270  85.283639 126.588561 171.20504
 [5,]  0.00000  0.00000  0.00000   0.000000   1.984035   3.8042231   7.2417271  13.666759  25.522622  47.045224  85.28495
 [6,]  0.00000  0.00000  0.00000   0.000000   0.000000   0.2989524   0.6218732   1.293605   2.690924   5.597592  11.64397
 [7,]  0.00000  0.00000  0.00000   0.000000   0.000000   0.0000000   0.0000000   0.000000   0.000000   0.000000   0.00000
 [8,]  0.00000  0.00000  0.00000   0.000000   0.000000   0.0000000   0.0000000   0.000000   0.000000   0.000000   0.00000
 [9,]  0.00000  0.00000  0.00000   0.000000   0.000000   0.0000000   0.0000000   0.000000   0.000000   0.000000   0.00000
[10,]  0.00000  0.00000  0.00000   0.000000   0.000000   0.0000000   0.0000000   0.000000   0.000000   0.000000   0.00000
[11,]  0.00000  0.00000  0.00000   0.000000   0.000000   0.0000000   0.0000000   0.000000   0.000000   0.000000   0.00000

$stree
       [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]    [,10]    [,11]
 [1,] 441.61 477.0126 515.2532 556.5596 601.1773 649.3719 701.4301 757.6617 818.4012 884.0100 954.8784
 [2,]   0.00 408.8412 441.6168 477.0199 515.2612 556.5681 601.1865 649.3819 701.4409 757.6733 818.4138
 [3,]   0.00   0.0000 378.5040 408.8475 441.6236 477.0272 515.2691 556.5767 601.1958 649.3919 701.4517
 [4,]   0.00   0.0000   0.0000 350.4178 378.5098 408.8538 441.6304 477.0346 515.2770 556.5853 601.2050
 [5,]   0.00   0.0000   0.0000   0.0000 324.4158 350.4232 378.5156 408.8601 441.6372 477.0419 515.2850
 [6,]   0.00   0.0000   0.0000   0.0000   0.0000 300.3431 324.4208 350.4286 378.5214 408.8664 441.6440
 [7,]   0.00   0.0000   0.0000   0.0000   0.0000   0.0000 278.0568 300.3478 324.4257 350.4340 378.5273
 [8,]   0.00   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000 257.4241 278.0611 300.3524 324.4307
 [9,]   0.00   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000 238.3225 257.4281 278.0653
[10,]   0.00   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000 220.6382 238.3261
[11,]   0.00   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000 204.2662
```

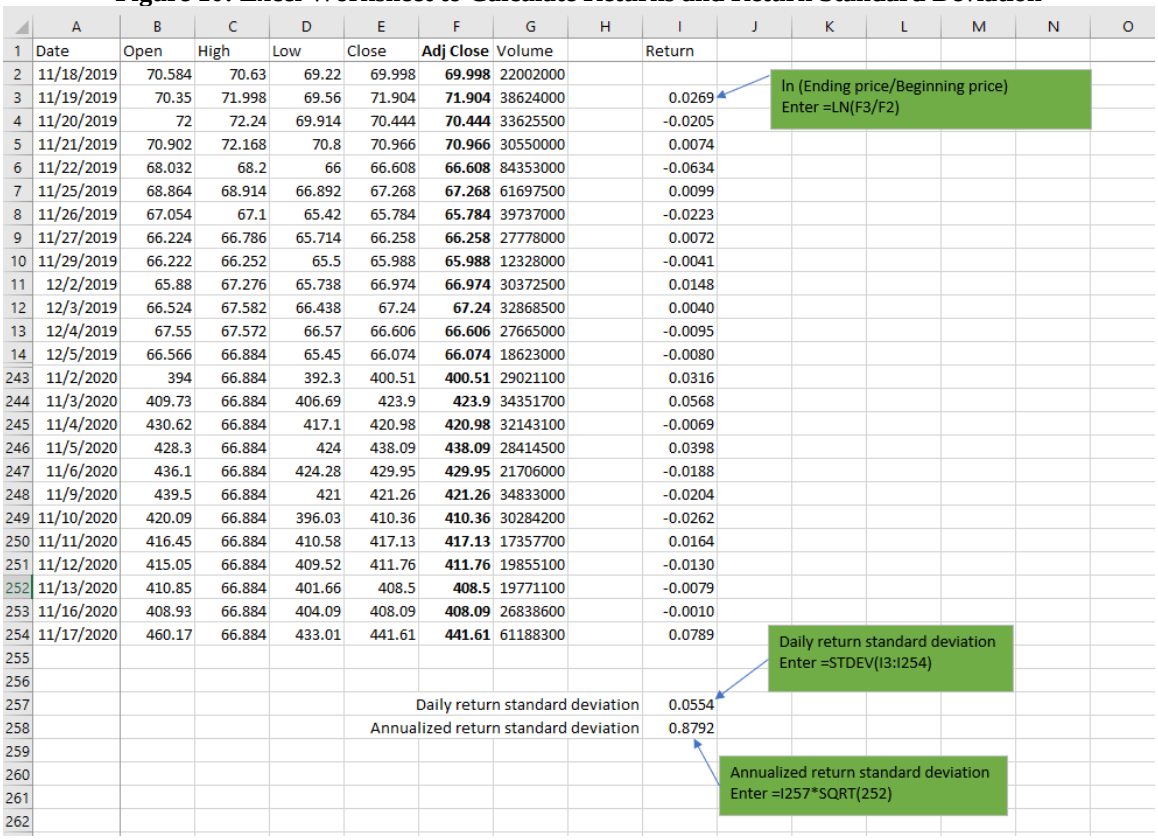
Using Excel in Option Pricing

In our R project, we also require students to use Excel to import data, calculate stock returns and return standard deviation, and price a European call option and a European put option of a stock they choose, using the Black-Scholes model and the multi-period binomial option pricing model. Using both R programming and Excel to price options help students solidify their understanding of the pricing models. Holden (2012) provides detailed illustrations on how to apply both the Black-Scholes model and the multi-period binomial option pricing model in Excel. Wann (2015) focuses on the Excel application of the Black-Scholes model, and Baril, Betancourt, and Briggs (2005) focus on the Excel application of the binomial (lattice) model. Since most of our students already have some experience with Excel, we focus on important Excel functions when we illustrate the Excel part of this project in class.

Electronic copy available at: https://ssrn.com/abstract=3895539

Figure 10 shows the calculation of the stock returns and return standard deviation, using the same non-dividend paying stock of Tesla as the example. After opening the CSV data file downloaded from Yahoo! Finance, we use the Excel function *"ln()"* to calculate the daily stock return based on the adjusted close price, and the function *"stdev()"* to calculate the daily return standard deviation. We obtain the annualized return standard deviation by multiplying the daily return standard deviation with the square root (*"sqrt()"*) of 252 (trading days per year). The annualized return standard deviation is 0.8792.

Figure 10: Excel Worksheet to Calculate Returns and Return Standard Deviation



	A	B	C	D	E	F	G	H	I
1	Date	Open	High	Low	Close	**Adj Close**	Volume		Return
2	11/18/2019	70.584	70.63	69.22	69.998	**69.998**	22002000		
3	11/19/2019	70.35	71.998	69.56	71.904	**71.904**	38624000		0.0269
4	11/20/2019	72	72.24	69.914	70.444	**70.444**	33625500		-0.0205
5	11/21/2019	70.902	72.168	70.8	70.966	**70.966**	30550000		0.0074
6	11/22/2019	68.032	68.2	66	66.608	**66.608**	84353000		-0.0634
7	11/25/2019	68.864	68.914	66.892	67.268	**67.268**	61697500		0.0099
8	11/26/2019	67.054	67.1	65.42	65.784	**65.784**	39737000		-0.0223
9	11/27/2019	66.224	66.786	65.714	66.258	**66.258**	27778000		0.0072
10	11/29/2019	66.222	66.252	65.5	65.988	**65.988**	12328000		-0.0041
11	12/2/2019	65.88	67.276	65.738	66.974	**66.974**	30372500		0.0148
12	12/3/2019	66.524	67.582	66.438	67.24	**67.24**	32868500		0.0040
13	12/4/2019	67.55	67.572	66.57	66.606	**66.606**	27665000		-0.0095
14	12/5/2019	66.566	66.884	65.45	66.074	**66.074**	18623000		-0.0080
243	11/2/2020	394	66.884	392.3	400.51	**400.51**	29021100		0.0316
244	11/3/2020	409.73	66.884	406.69	423.9	**423.9**	34351700		0.0568
245	11/4/2020	430.62	66.884	417.1	420.98	**420.98**	32143100		-0.0069
246	11/5/2020	428.3	66.884	424	438.09	**438.09**	28414500		0.0398
247	11/6/2020	436.1	66.884	424.28	429.95	**429.95**	21706000		-0.0188
248	11/9/2020	439.5	66.884	421	421.26	**421.26**	34833000		-0.0204
249	11/10/2020	420.09	66.884	396.03	410.36	**410.36**	30284200		-0.0262
250	11/11/2020	416.45	66.884	410.58	417.13	**417.13**	17357700		0.0164
251	11/12/2020	415.05	66.884	409.52	411.76	**411.76**	19855100		-0.0130
252	11/13/2020	410.85	66.884	401.66	408.5	**408.5**	19771100		-0.0079
253	11/16/2020	408.93	66.884	404.09	408.09	**408.09**	26838600		-0.0010
254	11/17/2020	460.17	66.884	433.01	441.61	**441.61**	61188300		0.0789
255									
256									
257						Daily return standard deviation			0.0554
258						Annualized return standard deviation			0.8792
259									
260									
261									
262									

Figure 11 shows the application of the Black-Scholes model to price the same European call option and European put option we define before. We use the Excel function *"normsdist()"*, the standard normal cumulative distribution function, to obtain the probability that a random draw from a standard normal distribution will be less than a specific number. The value of the European call option is $48.3612 and the value of the European put option is $36.7182, the same as what we obtain from using R programming.

Electronic copy available at: https://ssrn.com/abstract=3895539

Figure 11: Excel Worksheet for the Black-Scholes Model

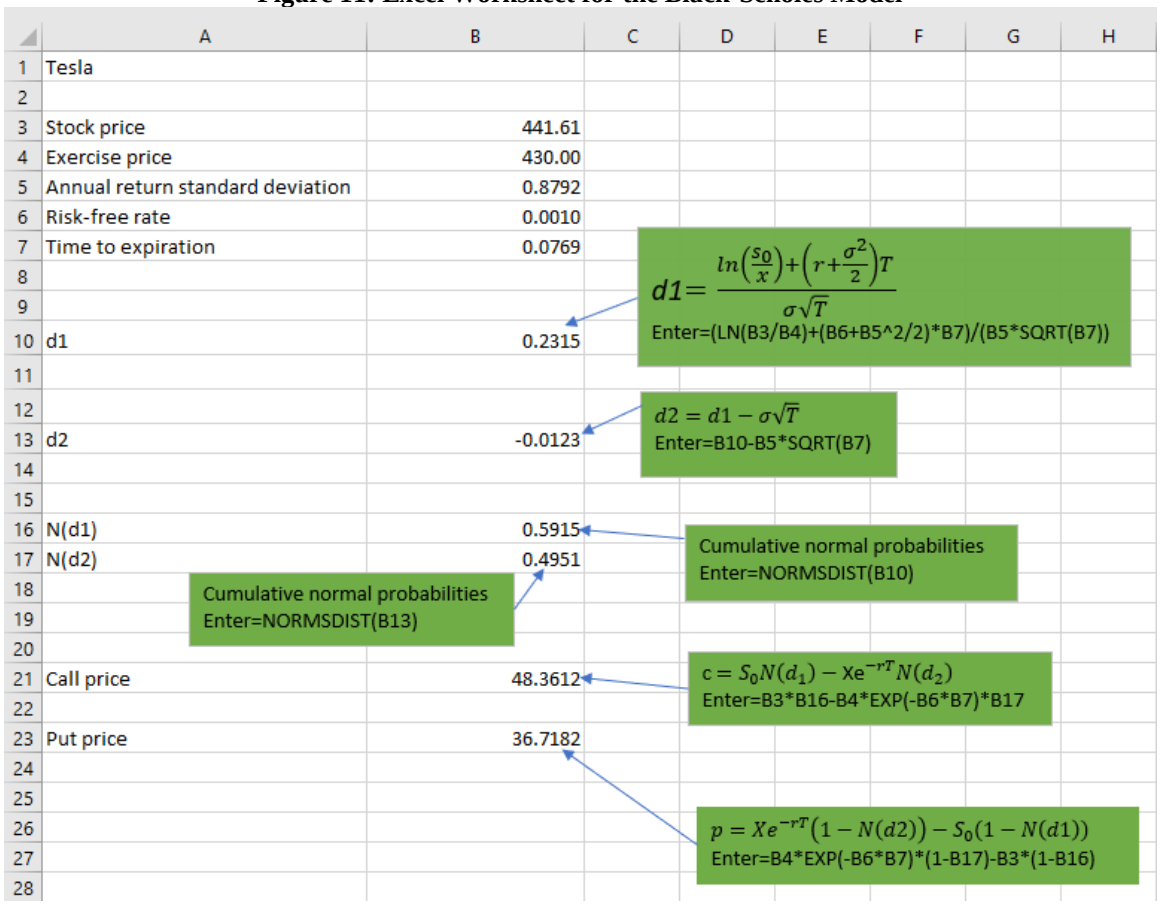
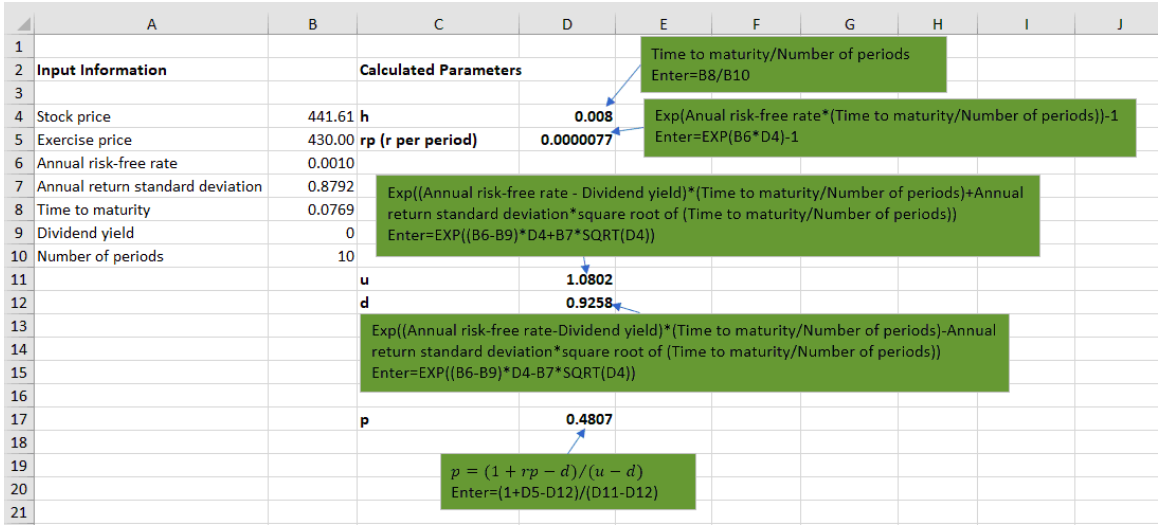


Figure 12 shows the parameter calculation for the multi-period binomial option pricing model. The definitions of the risk-free rate per period (*rp*), *u(p)*, *d(n)*, and the binomial (risk-neutral) probability (*p*) are the same as in the R package *"derivmkts"*, and their values are respectively 0.0000077, 1.0802, 0.9258, and 0.4807 in this example.

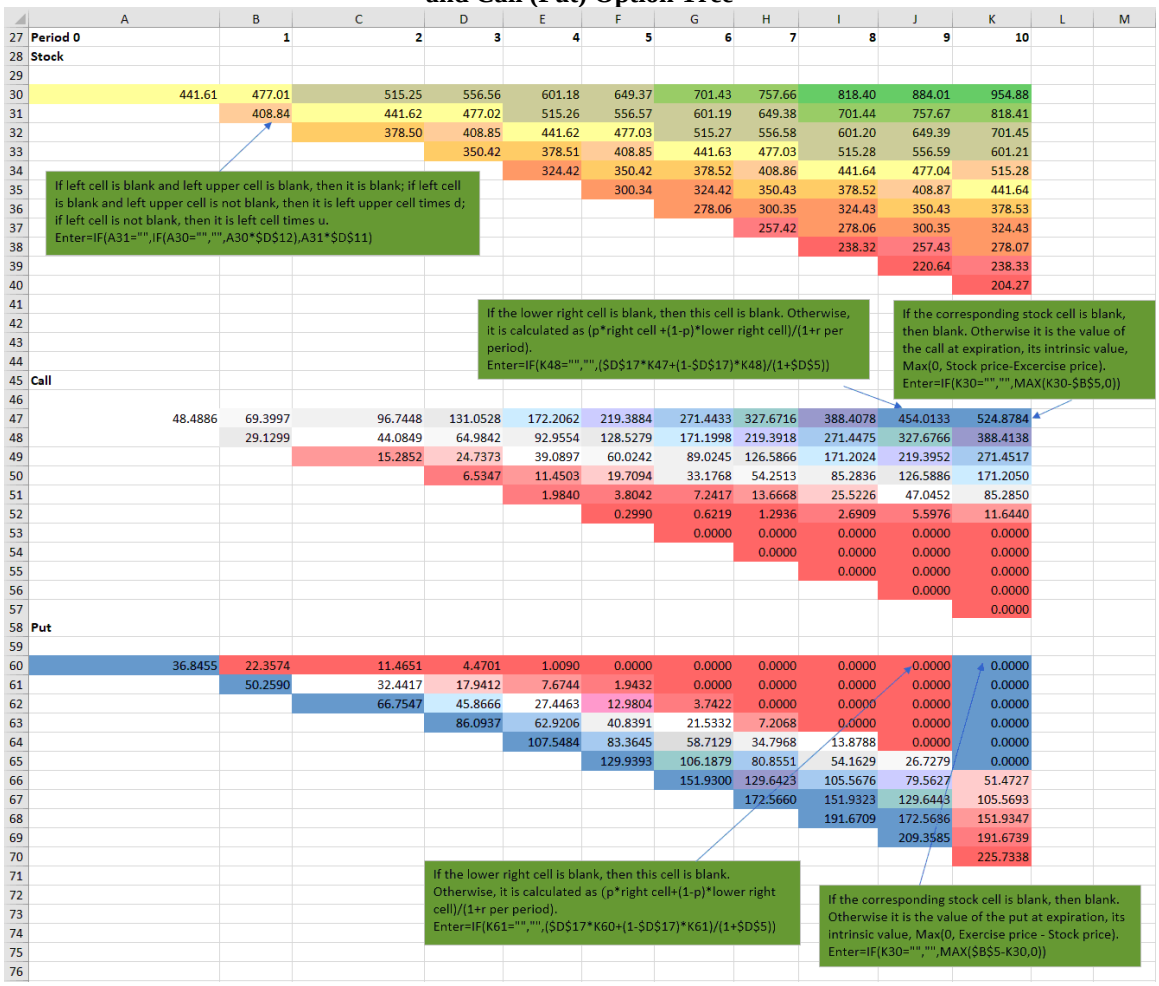
Figure 12: Excel Worksheet for the Multi-period Binomial Option Pricing Model – Parameter Calculation


Input Information
Stock price 441.61
Exercise price 430.00
Annual risk-free rate 0.0010
Annual return standard deviation 0.8792
Time to maturity 0.0769
Dividend yield 0
Number of periods 10
Calculated Parameters
h 0.008
rp (r per period) 0.0000077
u 1.0802
d 0.9258
p 0.4807
Time to maturity/Number of periods
Enter=B8/B10
Exp(Anual risk-free rate*(Time to maturity/Number of periods))-1
Enter=EXP(B6*D4)-1
Exp((Annual risk-free rate - Dividend yield)*(Time to maturity/Number of periods)+Annual return standard deviation*square root of (Time to maturity/Number of periods))
Enter=EXP((B6-B9)*D4+B7*SQRT(D4))
Exp((Annual risk-free rate-Dividend yield)*(Time to maturity/Number of periods)-Annual return standard deviation*square root of (Time to maturity/Number of periods))
Enter=EXP((B6-B9)*D4-B7*SQRT(D4))
$p = (1 + rp - d)/(u - d)$
Enter=(1+D5-D12)/(D11-D12)

Electronic copy available at: https://ssrn.com/abstract=3895539

Once we have the parameter values, we use the Excel functions *"if()"* and *"max()"* to construct the stock price tree and the call (put) option tree. The *"if()"* function performs a logical test on the content of a cell and returns a specific value depending on whether the result of the logical test is TRUE or FALSE. The *"max()"* function returns the maximum value among the inputs within the parentheses. Figure 13 shows the stock price tree, the call (put) option tree, and the derived values of the European call option and the European put option.

Figure 13: Excel Worksheet for the Multi-period Binomial Option Pricing Model – Stock Price Tree and Call (Put) Option Tree


If left cell is blank and left upper cell is blank, then it is blank; if left cell is blank and left upper cell is not blank, then it is left upper cell times d; if left cell is not blank, then it is left cell times u.
Enter=IF(A31="",IF(A30="","",A30*$D$12),A31*$D$11)
If the lower right cell is blank, then this cell is blank. Otherwise, it is calculated as (p*right cell +(1-p)*lower right cell)/(1+r per period).
Enter=IF(K48="","",($D$17*K47+(1-$D$17)*K48)/(1+$D$5))
If the corresponding stock cell is blank, then blank. Otherwise it is the value of the call at expiration, its intrinsic value, Max(0, Stock price-Excercise price).
Enter=IF(K30="","",MAX(K30-$B$5,0))
If the lower right cell is blank, then this cell is blank. Otherwise, it is calculated as (p*right cell+(1-p)*lower right cell)/(1+r per period).
Enter=IF(K61="","",($D$17*K60+(1-$D$17)*K61)/(1+$D$5))
If the corresponding stock cell is blank, then blank. Otherwise it is the value of the put at expiration, its intrinsic value, Max(0, Exercise price - Stock price).
Enter=IF(K30="","",MAX($B$5-K30,0))

Using the calculated parameters, we first construct the 10-period (step) stock price tree shown in Figure 13 as follows. We start with the current stock price at cell A30, then set up the function in cell B31 and copy the function to the other cells to obtain the 10-period binomial stock price tree. Once we have the stock price tree, we move backwards from period 10 to period 0 to construct the call (put) option tree. Specifically, we use both the *"if()"* and *"max()"* functions in cell K47 (K60), the first cell of the call (put) option tree in period 10, to get the option value at expiration, and copy the function to the other period-10 cells. We then use the *"if()"* function in cell J47 (J60) to get the call (put) option value in the first cell of period 9, and copy the function to the other period-9 cells. We repeat the same procedure backwards for other periods until we reach period 0 with the option value. In this case the value of the European call option is $48.4886 and the value of the European put option is $36.8455, the same as what we obtain from using R programming.

Overall we find both procedures in R programming and in Excel to price European options using the Black-Scholes model easy to implement. Using the multi-period binomial option model, however, R programming with the third-party package has the advantage of being more flexible and accommodating. For instance, it is time-consuming to construct a 100-period stock price tree and option trees in Excel. In R

Electronic copy available at: https://ssrn.com/abstract=3895539

programming with the third-party package, moving from a 10-period tree to a 100-period tree involves only one change in the package parameters. R programming is thus more efficient than Excel in handling complicated problems.

Discussion

Our goal of incorporating the R project into the undergraduate financial derivatives course is to expose students to the popular R programming language. After the successful completion of this project, students gain experience in importing and manipulating data in R, using R to do calculations, constructing user-defined functions, and applying third-party packages in R. The R project not only allows our students to pursue more advanced applications using R, but also allows us to maintain the currency of our curriculum. For instance, the Association to Advance Collegiate Schools of Business (AACSB International) recently issued the 2020 Business Accreditation Standards (AACSB International 2020a) and the associated Interpretive Guidance (AACSB International 2020b). Among the standards, Standard 4.1 on Curriculum (Content) requires that "[t]he school delivers content that is current, relevant, forward-looking, globally-oriented, aligned with program competency goals, and consistent with its mission, strategies, and expected outcomes. The curriculum content cultivates agility with current and emerging technologies." In addition, the Interpretive Guidance on Curriculum Content states that "[t]he purpose of this requirement is to demonstrate that schools are providing learners with relevant technology competencies in line with what might be expected for business degree graduates." Using the R programming language in the financial derivatives course through the class project thus maintains our curriculum currency by incorporating relevant technology into the curriculum.

Over the past few years in the financial derivatives course, we obtain overwhelmingly positive student responses to the R project through our class surveys. Students seem to be surprised by the fact that, they too, can write codes using the R programming language, and they have enjoyed the experience with the project. A few comments suggest that some students prefer a deeper coverage of R, which we find difficult to implement given the limited amount of time in a single course. Coordinated efforts through different courses, however, could possibly provide a deeper and more comprehensive coverage of R through the curriculum.

Conclusion

R is a popular and free programming language and software environment for statistical computing and graphics. Its open-source nature, compatibility with different operating systems, and the abundance of third-party packages together distinguish R from other programming languages. In this paper we illustrate in detail a class project in an undergraduate financial derivatives course that we use to introduce students to R programming. Specifically, through pricing options using the Black-Scholes model and the multi-period binomial option pricing model, students learn to import and manipulate data in R, use R to do calculations, construct user-defined functions, and apply third-party packages in R. The project equips our students with the knowledge of R for more advanced applications, and allows us to maintain the currency of our curriculum.

The R project discussed in this paper can be used directly in an undergraduate financial derivatives course. Components of the project, however, can also be used in other finance courses such as investments and corporate finance, and courses in other disciplines that have a need for data analysis. Students equipped with the knowledge of the R programming language surely will have a competitive advantage on the job market over those without.

References

AACSB International. 2020a. *2020 Guiding Principles and Standards for AACSB Business Accreditation*. *https://www.aacsb.edu/accreditation/standards/business*.

AACSB International. 2020b. *2020 Interpretive Guidance for AACSB Business Accreditation*. *https://www.aacsb.edu/accreditation/standards/business*.

Allen, David E. 2017. "Practical Aspects of R in Finance, Management Information and Decision Sciences." *Journal of Management Information and Decision Sciences* 20: 1-10.

Electronic copy available at: https://ssrn.com/abstract=3895539

Baril, Charles P., Luis Betancourt, and John W. Briggs. 2005. "How to "Excel" at Options Valuation." *Journal of Accountancy* 200: 57-63.

Black, Fisher, and Myron Scholes. 1973. "The pricing of Options and Corporate Liabilities." *Journal of Political Economy* 81: 637-654.

Clemmensen, Peter. 2017. "Black Scholes Price in SAS." Retrieved from *https://sasnrd.com/sas-black-scholes-price/*.

Das, Sanjiv Ranjan. 2014. "Text Extraction." *Foundations & Trends in Finance* 8: 150-167.

Dishman, Lydia. 2016. "Why Coding Is Still the Most Important Job Skill of the Future." *https://www.fastcompany.com/3060883/why-coding-is-the-job-skill-of-the-future-for-everyone*.

Gallagher, Michael, and Rossen Trendafilov. 2018. "R vs. Python: Ease of Use and Numerical Accuracy." *Journal of Business and Accounting* 11: 117-126.

Holden, Craig W. 2012. *Excel Modeling in Investments* (4th Edition). Pearson Education, Inc.

Krotov, Vlad, and Matthew Tennyson. 2018. "Research Note: Scraping Financial Data from the Web Using the R Language." *Journal of Emerging Technologies in Accounting* 15: 169-181.

Lei, Adam Y.C., and Huihua Li. 2012, "Using Bloomberg Terminals in a Security Analysis and Portfolio Management Course." *Journal of Economics and Finance Education* 11: 17-33.

McDonald, Robert. 2019. "Package 'derivmkts' Functions and R Code to Accompany Derivatives Markets." *https://cran.r-project.org/web/packages/derivmkts/derivmkts.pdf*.

Merton, Robert C. 1973. "Theory of Rational Option Pricing." *Bell Journal of Economics and Management Science* 4: 141-183

Muenchen, Robert A. 2020. "The Popularity of Data Science Software." *http://r4stats.com/articles/popularity/*.

Ozgur, Ceyhun, Sanjeev Jha, Elyse "Bennie" Myer-Tyson, and David Booth. 2018. "The Usage of R Programming in Finance and Banking Research." *Journal of Accounting and Finance* 18: 61-69.

Sedgewick, Robert, and Kevin Wayne. 2017. "BlackScholes.java." *https://introcs.cs.princeton.edu/java/22library/BlackScholes.java.html*.

The R Foundation. 2020. "What is R?" *https://www.r-project.org/about.html*.

Turner, James A. 2015. "Optimal Portfolios and the R Programming Language." *International Journal of the Academic Business World* 9: 47-57.

de Vries, Andrie, and Joris Meys. 2015. *R for Dummies* (2nd Edition). John Wiley & Sons, Inc.

Wann, Christi. 2015, "Black-Scholes Option Pricing: Implementing a Hands-On Assignment Using Excel." *Journal of Economics and Finance Education* 14: 22-30.

Wikipedia. 2020. "R(programming_language)." *https://en.wikipedia.org/wiki/R_(programming_language)*.

Zagaglia, Paolo. 2021. "Option pricing package." *https://www.mathworks.com/matlabcentral/fileexchange/35351-option-pricing-package*.

Electronic copy available at: https://ssrn.com/abstract=3895539

APPENDIX: R FUNCTIONS USED

This appendix summarizes the R functions used in this paper. Details of these functions are available through the Help function of RStudio.

as.Date(): date conversion function to and from character.
c(): combine values into a vector or list.
cbind(): combine R objects by rows or columns.
exp(): calculate the power of *e.*
head(): returns the first or last part of a R object.
install.packages(): download and install packages from CRAN-like repositories or local files.
library(): load packages.
list(): create a list of R objects.
log(): calculate the natural logarithm.
pnorm(): return the probability under the cumulative normal distribution function.
read.cvs(): import a CSV file and create a data frame.
sd(): calculate the standard deviation.
sqrt(): calculate the square root.
str(): display the structure of a R object.
View(): show the content of a data frame.

Electronic copy available at: https://ssrn.com/abstract=3895539